R07922141

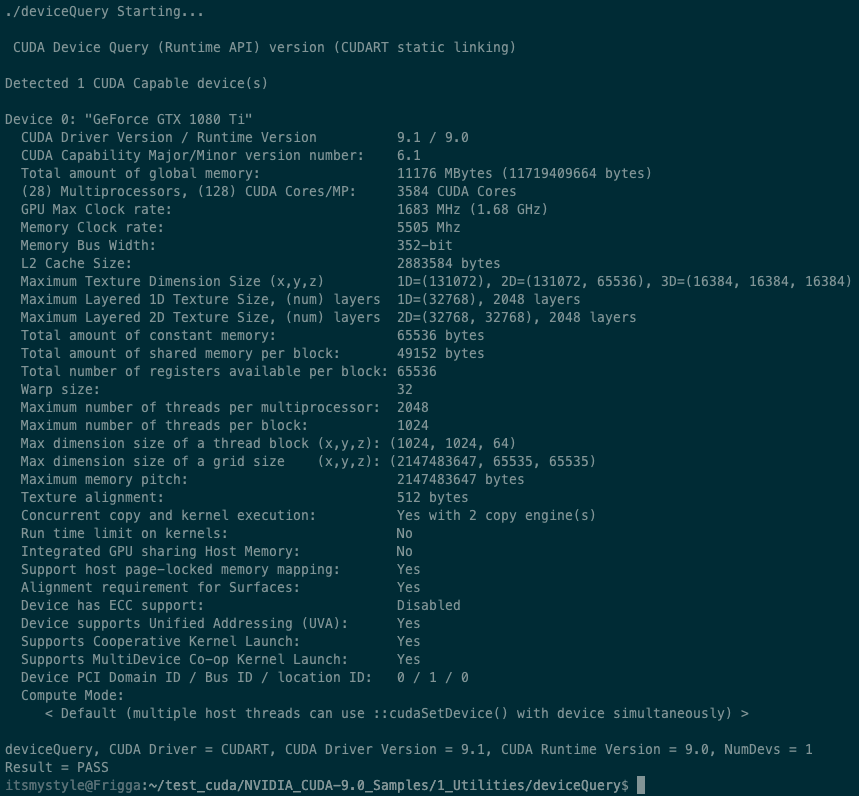
張緣彩

31/10/2018

Data Mining Hw2 Report

**I use my own code from homework 1.**

**Part I. Setting up CUDA environment**



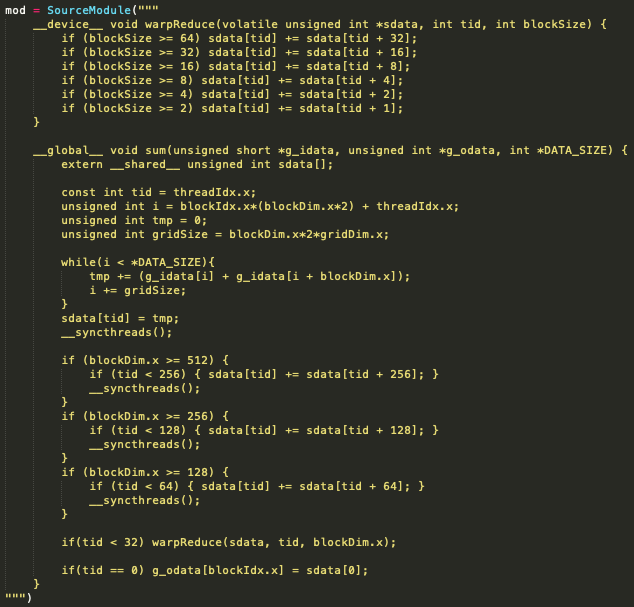
**Part II. Frequent Itemset Mining with GPGPU**

**Implementation**

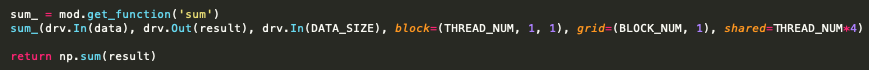
Since the Eclat algorithm has not change much compare to my homework 1, I will not explain much in Eclat algorithm. The only different is the summation function, I have changed it to GPU summation. For more information about Eclat algorithm, please refer to my homework 1 report. I will mainly focus on explaining my PyCuda kernel function.

First, I change the np.sum() to my gpusum(). Which input arguments are data, Data size, Block number and Thread number.

For my parallel kernel, I implemented the parallel summation with reduction techniques according to the hint. But there is a bit problem in the hint sample code. If data size is larger than block number \* thread number \* 2, we must add multiple rounds to sdata, but I found that there is synchronize problem when performing save data into sdata. So I decided to add the input data and save it to a temporary memory, then move the temporary into sdata at the end of while loop.



Since the input data is a bitvector which is a list of 1 and 0, so I decided to make the cpu-gpu IO faster with change integer to unsigned short. I remove the template and change the BlockSize to blockDim.x.

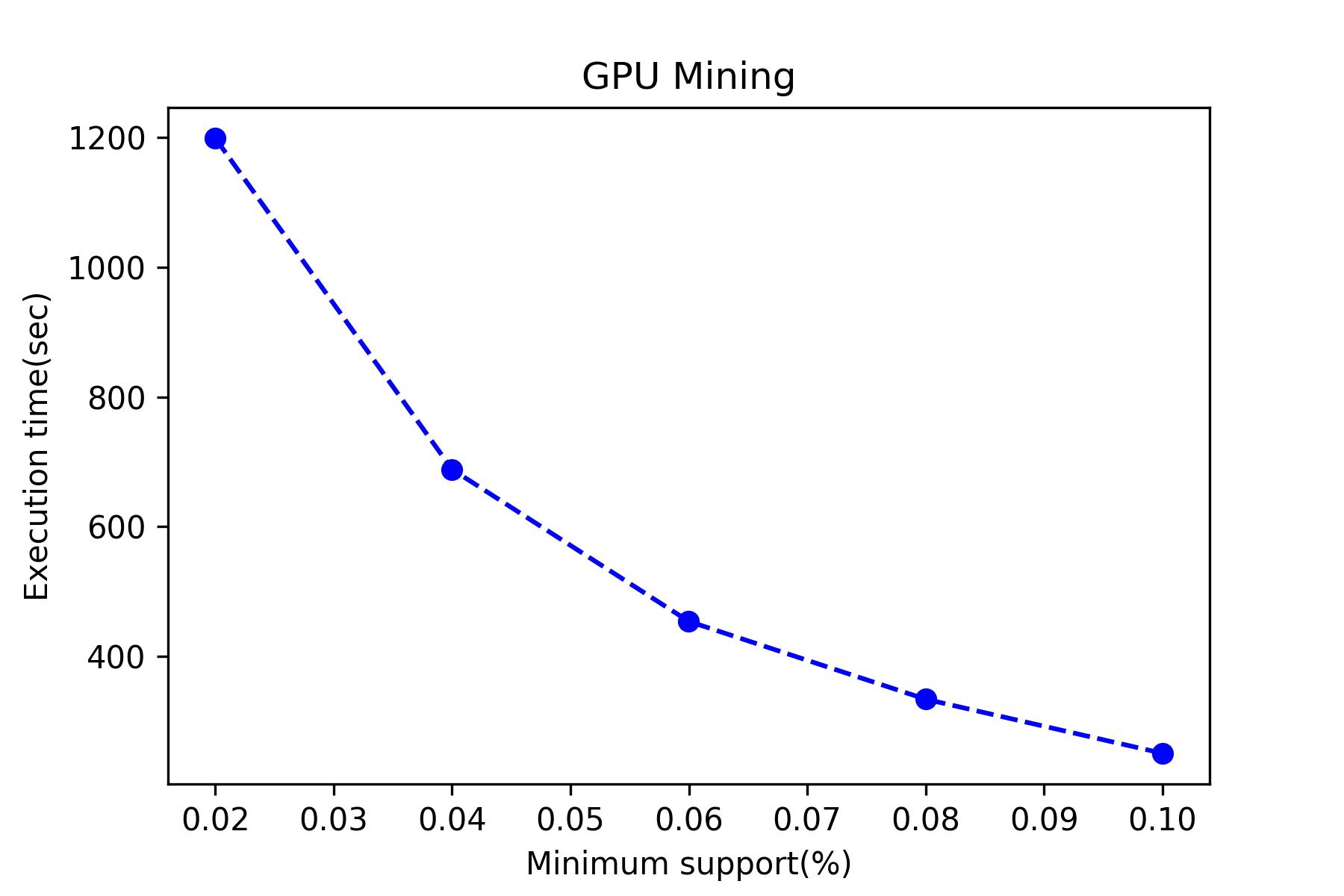


The finally return the result over numpy summation.

**Graph**

* **Different minimum support. (Thread number 128, Block number 128)**

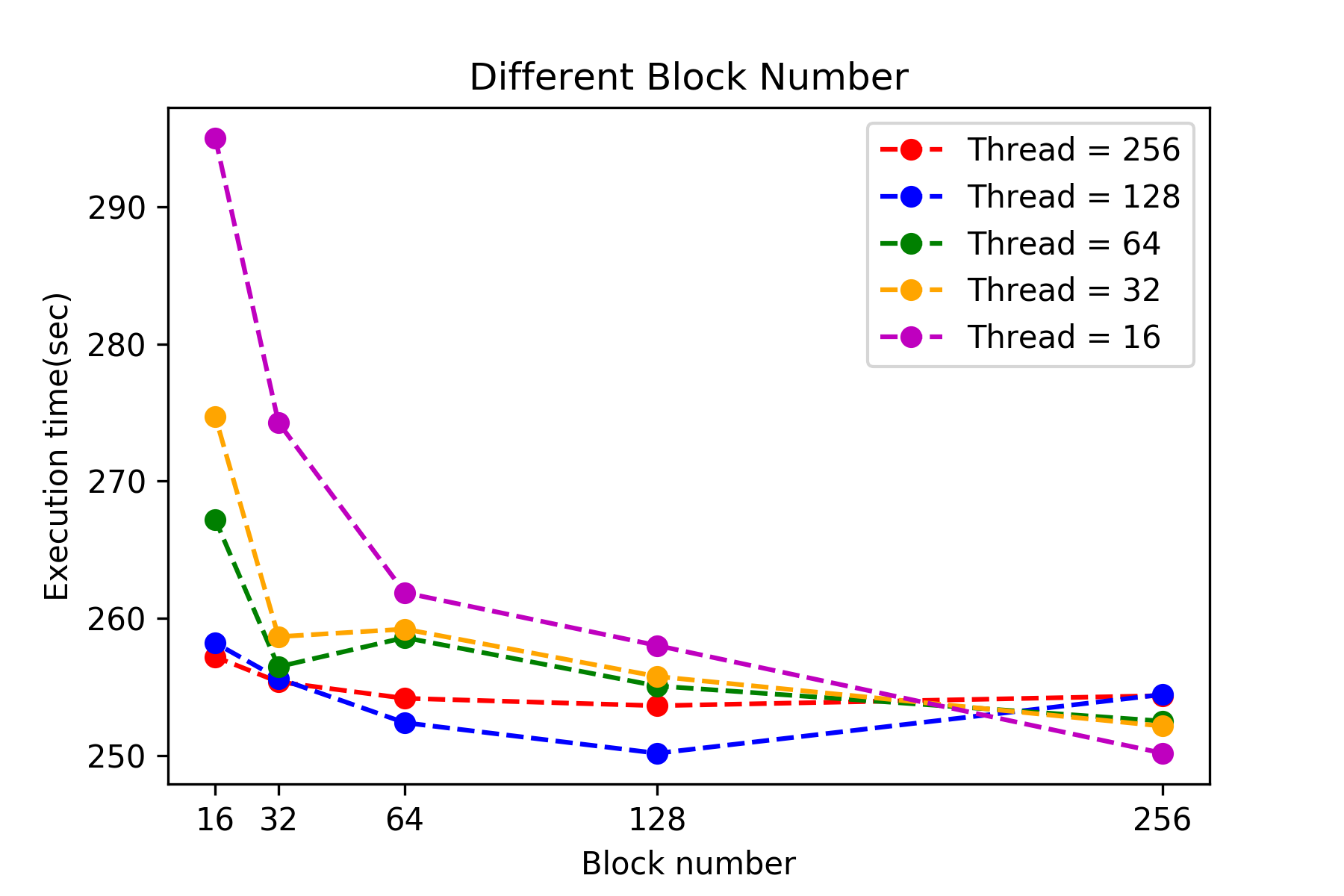
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Min sup. | 0.1% | 0.08% | 0.06% | 0.04% | 0.02% |
| Time(s) | 250.151 | 333.610 | 453.652 | 688.114 | 1199.028 |



The figure above showed us the execution time with different minimum support. The execution time and minimum support have positive correlation. This is because the smaller the minimum support, the more itemset that GPU has to operate.

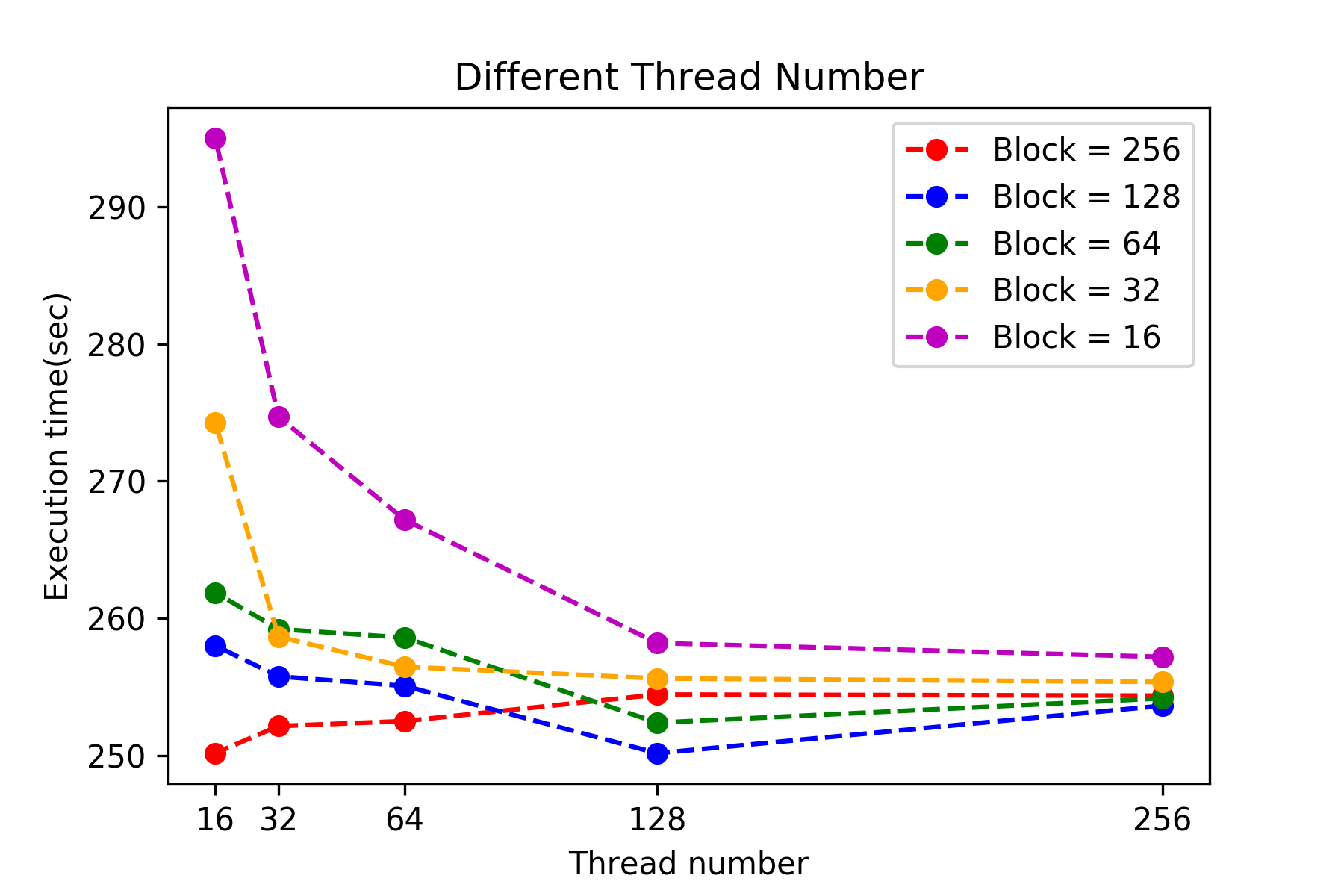
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T/B | 256 | 128 | 64 | 32 | 16 |
| 256 | 254.351 | 253.618 | 254.158 | 255.354 | 257.182 |
| 128 | 254.434 | 250.151 | 252.379 | 255.607 | 258.187 |
| 64 | 252.498 | 255.059 | 258.585 | 256.461 | 267.181 |
| 32 | 252.133 | 255.752 | 259.207 | 258.654 | 274.687 |
| 16 | 250.162 | 258.007 | 261.863 | 274.272 | 295.005 |

* **Different block and thread number. (Minimum support 0.1%)**
  + **Different block number.**



The figure above described the execution time of different block number. With all five lines, we can conclude that the execution time decreased over block number increment. Especially the purple line (thread = 16), the execution time decrease rapidly according to block number because the many the block are, the many the threads can be operating in the same time.

* + **Different thread number.**



The figure above described the execution time of different thread number. We can conclude that the execution time decreased over thread number increments. Especially for the purple line (block = 16), the execution time decrease rapidly according to thread number because the many the thread are, the more the operation power can be operate in the same time.